Natural language understanding with transformers

Lecture slides

Outline

- Typology of NLU tasks
- Sentiment analysis
- Natural language inference
- Named entity recognition
- Question answering

Types of NLU problems

Types of NLU problems

NLU = natural language understanding = extracting meta data from texts

Single text classification
- spam detection
- sentiment analysis
- tagging articles by topic
- toxic comment identification
- intent classification in dialogue systems

Token classification
- named entity recognition (NER)
- part of speech tagging
- slot filling in dialogue systems
- toxic spans detection
- in-text question answering
- grammatical errors detection (and sometimes correction)

Relations between texts
- paraphrase detection
- natural language inference (NLI)
- matching documents with queries
- translation identification
- multiple choice question answering

Relations between tokens
- dependency or constituency parsing
- coreference resolution
- extraction of relations between entities
- nested named entity recognition

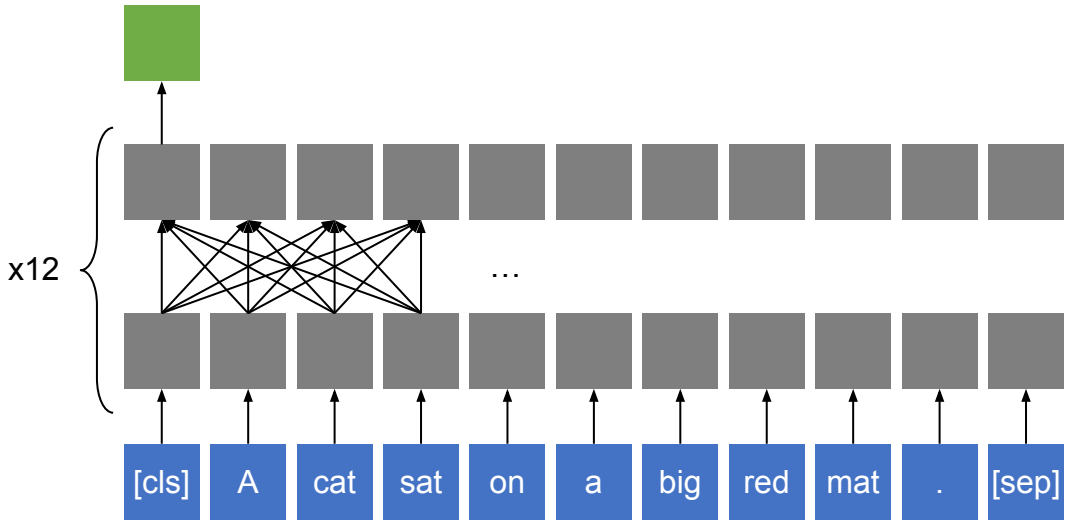
NLP problems NOT related to NLU

- Text-to-text (sequence-to-sequence = seq2seq) problems
  - translation
  - summarization
  - dialogue response generation
  - generative question answering
- Text-to-structure problems: seq2seq with a special output format
  - text to code (e.g. SQL, SPARQL, various programming languages)
  - text to meaning representation (e.g. AMR)
- These problems typically require text decoders
  - We will consider them next time
  - Today, we focus on the problems that can be solved with an encoder-only model (such as BERT)

Architectures for typical NLU problems

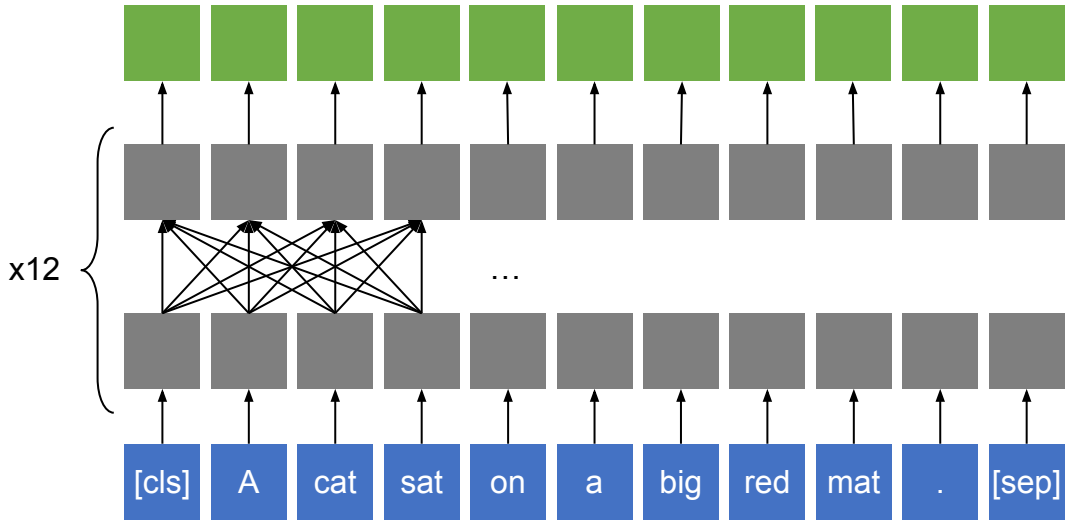
BERT* for sequence classification

- Add a linear layer on top of **the first token**
- Fine-tune the whole model
  - The model will automatically learn to deliver all necessary information to the first token



BERT* for token classification

- Add a linear layer on top of **each token**
  - Use the same weights for each token
- Fine-tune the whole model

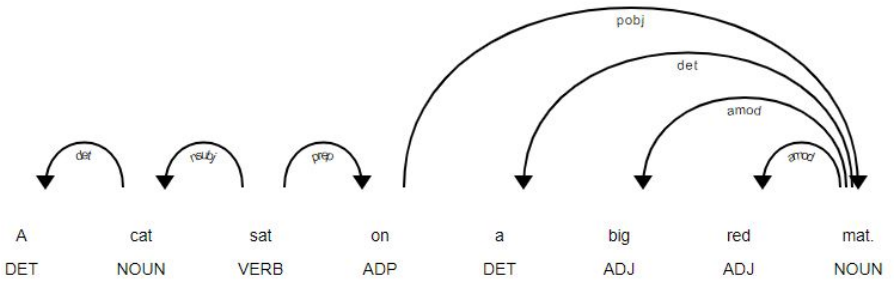


* Instead of BERT, any other pretrained transformer encoder can be applied

Biaffine layers for relations between tokens

The goal: predict pairwise relations between tokens

- Use a binary classifier for each pair of tokens (or multiclass, to predict relation type)
- Biaffine layer is a simple way to combine two vectors



	A	cat	sat	on	A	big	red	mat
A	0	1	0	0	0	0	0	0
cat	0	0	1	0	0	0	0	0
sat	0	0	0	0	0	0	0	0
on	0	0	1	0	0	0	0	0
a	0	0	0	0	0	0	0	1
big	0	0	0	0	0	0	0	1
red	0	0	0	0	0	0	0	1
mat	0	0	0	1	0	0	0	0

Biaffine function for predicting the (i, j) arc:

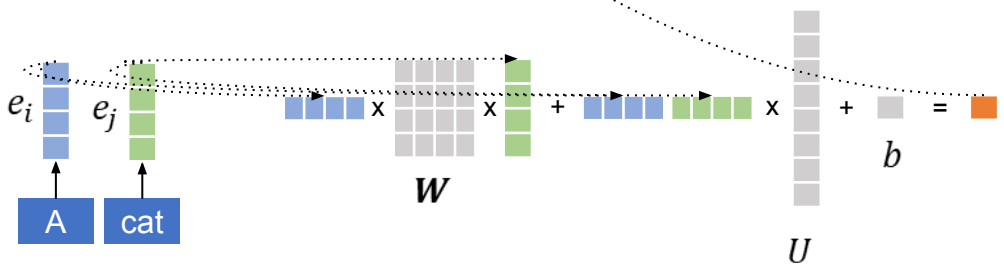
$$score_{ij} = e_i^T \boldsymbol{W} e_j + \left(e_i \oplus e_j\right)^T U + b$$

e_i, e_j – contextual token embeddings

Parameters: $\boldsymbol{W}$ – a matrix, U – a vector, b – a scalar

"Affine" means "linear with bias".

The function above is affine w.r.t. each argument, i.e. biaffine.



Source: Deep Biaffine Attention for Neural Dependency Parsing, Dozat et al, 2017

Sentiment analysis

Sentiment analysis

- "Sentiment analysis" is an umbrella term for multiple tasks:
  - Predict general polarity of a text (positive/neutral/negative)
    - *Coronet has the best lines of all day cruisers.*
    - *Pastel-colored 1980s day cruisers from Florida are ugly.*
  - Predict polarity w.r.t. a specific object of its *aspect*
    - *I was tempted to buy this product as I really like its design, but its price is not good.*
    - *Chris Craft is better looking than Limestone, but Limestone projects seaworthiness and reliability.*
  - Predict fine-grained emotions (joy/sadness/fear/anger/surprise etc.)
- Diverse levels of difficulty
  - In simple cases, lists of keywords suffice
  - In complex cases, one has to deal with negations, collocations, sarcasm etc.

Sources: Stanford cs224u course, Wikipedia

Related classification problems

There are multiple other emotional phenomena that occur in natural texts and can be detected.

Their degree of difficulty is diverse.

- Subjectivity (Pang and Lee 2008)
- Bias (Recasens et al. 2013; Pryzant et al. 2020)
- Stance (Anand et al. 2011)
- Hate-speech (Nobata et al. 2016)
- Microaggressions (Breitfeller et al. 2019)
- Condescension (Wang and Potts 2019)
- Sarcasm (Khodak et al. 2017)
- Deception and betrayal (Niculae et al. 2015)
- Online trolls (Cheng et al. 2017)
- Polarization (Gentzkow et al. 2019)
- Politeness (Danescu-Niculescu-Mizil et al. 2013)
- Linguistic alignment (Doyle et al. 2016)

Source: Stanford cs224u course

Tips for sentiment analysis (and similar problems)

- *AutoModelForSequenceClassification* will do the job well
- Sometimes, up-sampling less frequent classes helps
  - Of course, it depends on the choice of metrics
- Both multiclass classification and regression work well
  - To convert between different sentiment scales, an ordered logistic regression head may be used
  - Sometimes, extra classes like “mixed sentiment” are required
- Data augmentation may help a lot
  - Randomly remove “too easy” features (e.g. punctuation, smileys)
  - Check how prediction change under simple transformations*

* A good example of such methodology is in Beyond Accuracy: Behavioral Testing of NLP models with CheckList by Ribeiro et al (2020)

Aspect-based sentiment analysis

- Analyze “Great food but the service is dreadful”
- Aspect extraction: given a text, detect “food” and “service”
  - Multilabel text classification: for each possible aspect (from a fixed set), predict whether it is present in the text
  - Text pair classification (NLI-style): given an aspect and a text, predict whether it is relevant (or even its sentiment)
  - Token classification (NER-style): highlight tokens that describe aspects
- Opinion extraction: given a text and an aspect, classify opinion
  - Single text classification, with aspect in special tags
  - Text pair classification (NLI-style)

Source: https://arxiv.org/abs/1910.00883, https://arxiv.org/abs/2011.00169

Natural language inference

Natural language inference

The goal of NLI: detect logical relation between two texts

Premise	Relation	Hypothesis
A turtle danced.	entails	A turtle moved.
turtle	contradicts	linguist
Every reptile danced.	neutral	A turtle ate.
Some turtles walk.	contradicts	No turtles move.
James Byron Dean refused to move without blue jeans.	entails	James Dean didn't dance without pants.
Mitsubishi Motors Corp's new vehicle sales in the US fell 46 percent in June.	contradicts	Mitsubishi's sales rose 46 percent.
Acme Corporation reported that its CEO resigned.	entails	Acme's CEO resigned.

Source: Stanford CS224U course, slides.

Why NLI is important

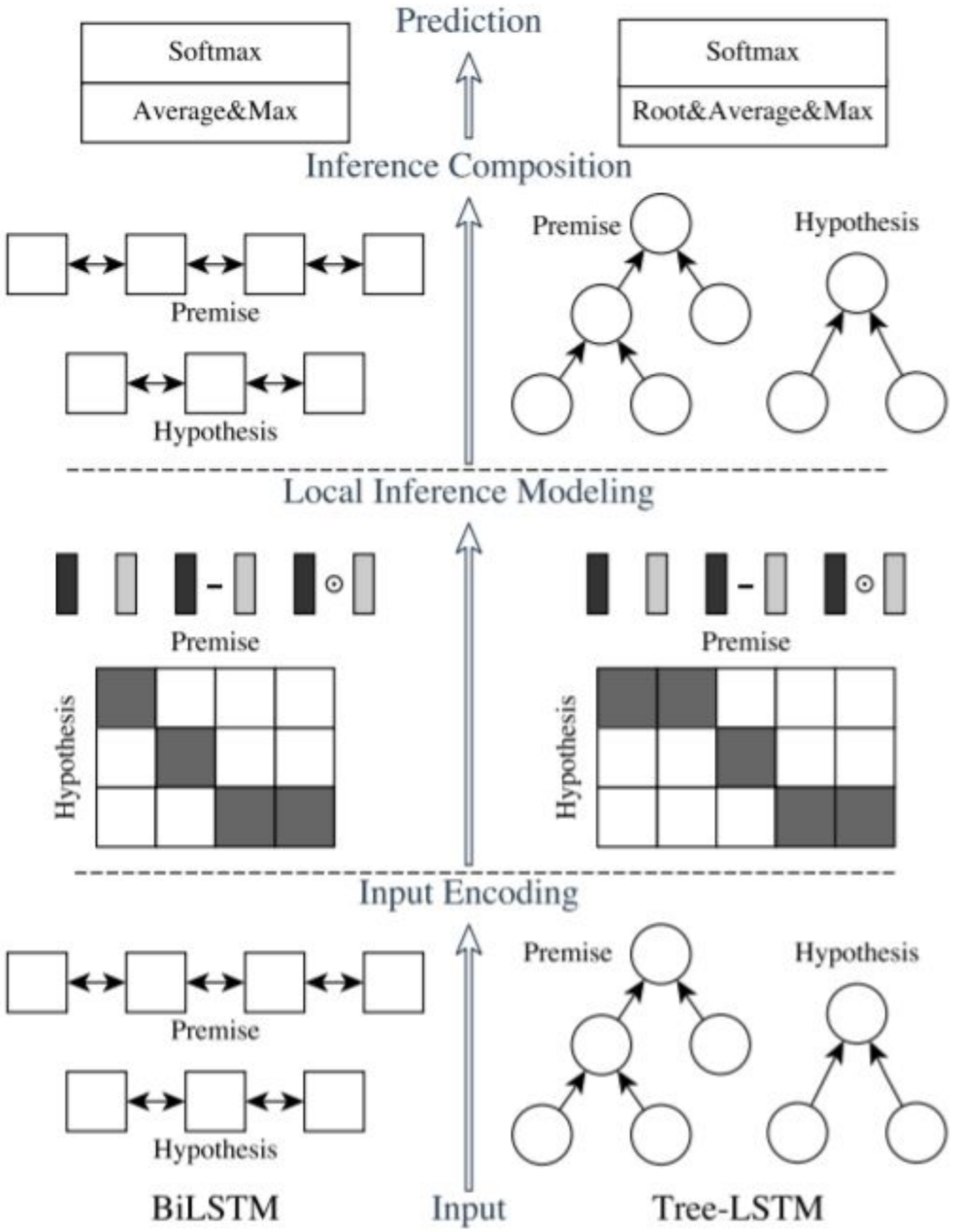
- Diagnostics
  - Evaluate a model's understanding of various linguistic phenomena, logic and world knowledge
    - For example, such a task is a part of the SuperGLUE benchmark
- Applications
  - Controlling quality of generated texts in various tasks
  - Paraphrase detection
  - Zero-shot or few-shot text classification
    - Does *"The IAU downgrade Pluto as a dwarf planet"* imply *"This is science news"*?
  - Pre-training for similar tasks

Task	**NLI framing**
Paraphrase	text ≡ paraphrase
Summarization	text ⊏ summary
Information retrieval	query ⊐ document
Question answering	question ⊐ answer
	Who left? ⇒ *Someone left*
	Someone left ⊐ *Sandy left*

Sources: Stanford CS224U course (slides); Wang et al, 2021, Entailment as Few-Shot Learner; https://gluebenchmark.com/diagnostics

NLI before transformers

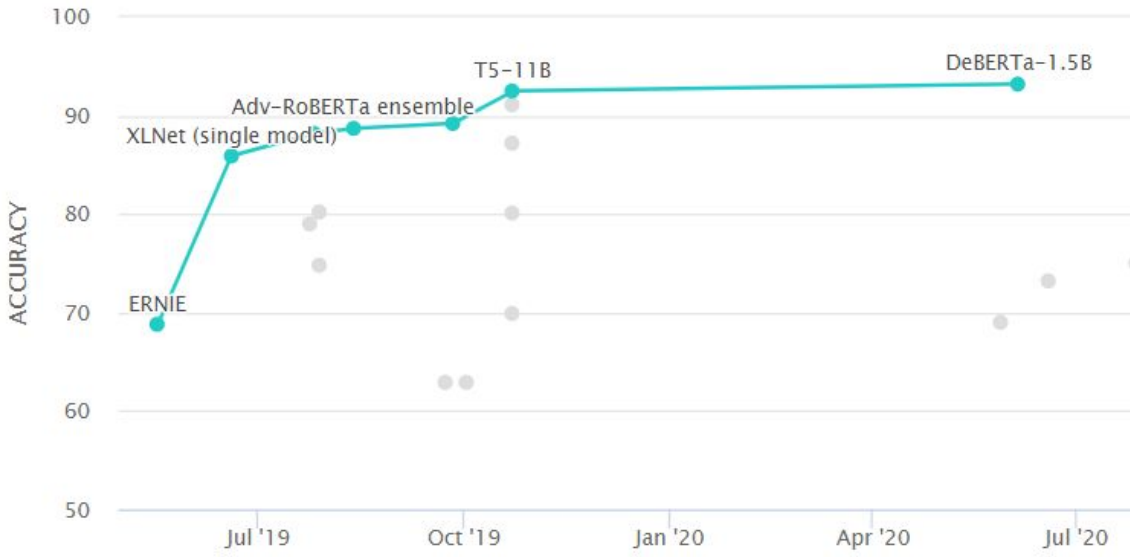
- Before transformers, NLI required special architectures, such as this one:
  - Compute contextual embeddings
  - Compute local interactions between each pair of terms in two texts
    - E.g. use a kind of attention
  - Update local embeddings using the information from interactions
  - Use pooling to get a fixed size vector
  - Predict the result with a MLP
- Transformers do all of this “naturally”



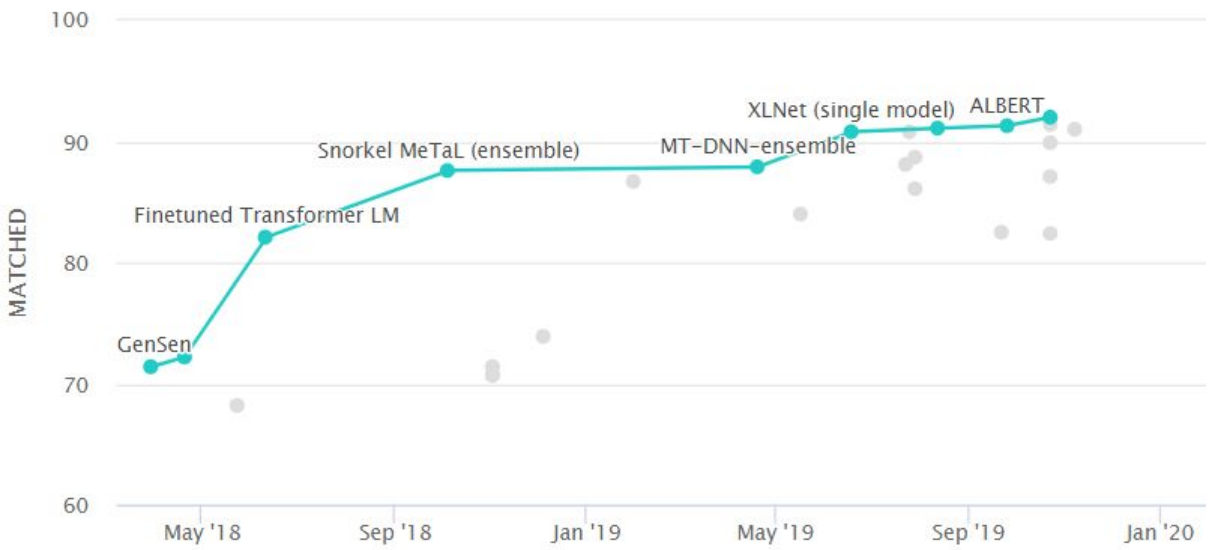
Chen et al, 2017, *Enhanced LSTM for Natural Language Inference*

NLI benchmarks

RTE (part of GLUE and SuperGLUE)



MultiNLI



- Transformers perform significantly better than alternative architectures
- Traditional benchmarks saturate fast

Source: https://paperswithcode.com/task/natural-language-inference

Adversarial NLI: a moving target

If standard NLI benchmarks are to easy, on can create an intentionally difficult dataset, using human-in-the-loop data collection

Repeat:

- Ask humans to create examples on which the current model makes mistakes
- Train a new model using these new difficult examples

Results:

- New difficult datasets
- Improvement even on the old datasets

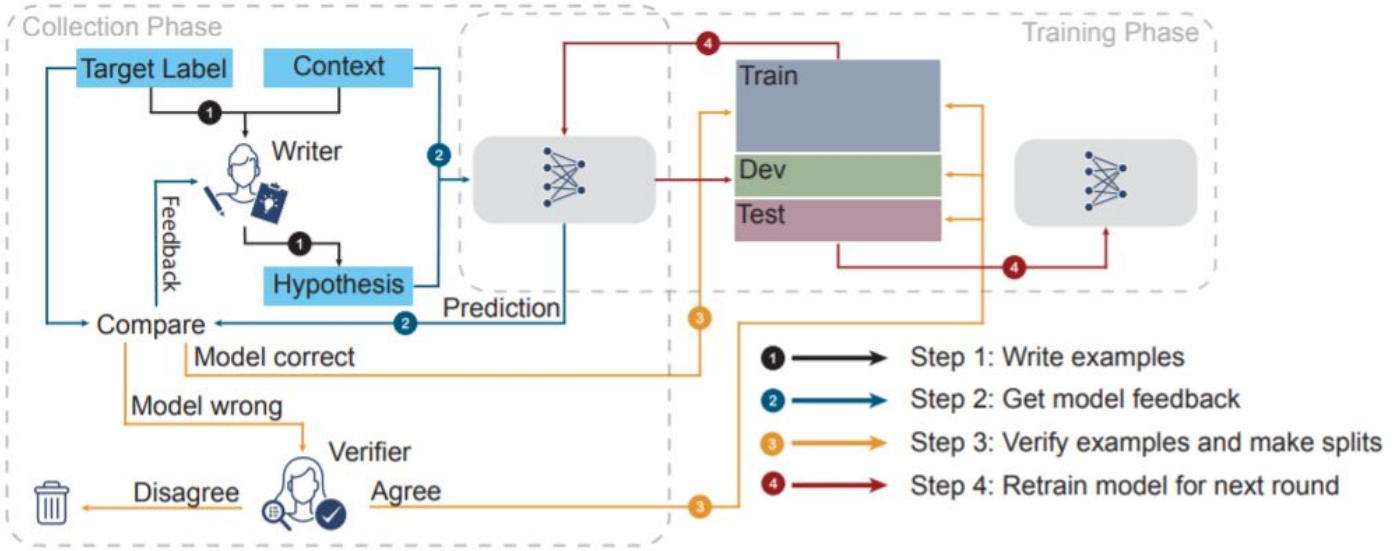


Figure 1: Adversarial NLI data collection via human-and-model-in-the-loop enabled training (HAMLET). The four steps make up one round of data collection. In step 3, model-correct examples are included in the training set; development and test sets are constructed solely from model-wrong verified-correct examples.

Nie et al, 2020, Adversarial NLI: A New Benchmark for Natural Language Understanding; https://dynabench.org/tasks/nli

Named entity recognition

Named entity recognition

- NER is the problem of finding named entities* within a text

- Typically, IO, BIO or BIOES schemes are used for labelling each token
  - BIO helps to distinguish adjacent entities
  - BIOES provides additional information (redundant but sometimes helpful) about entity boundaries

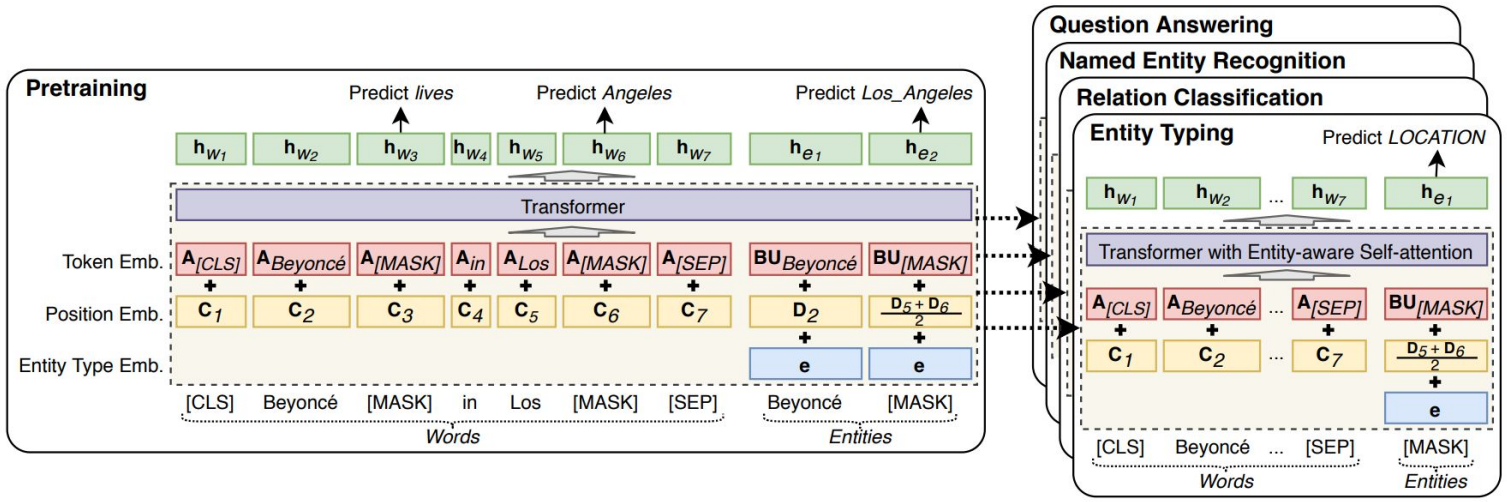
	Alexander	Sergeyevich	Pushkin	was	born	in	Moscow	in	1799	year
IO	PER	PER	PER	O	O	O	GPE	O	DATE	DATE
BIO	B-PER	I-PER	I-PER	O	O	O	B-GPE	O	B-DATE	I-DATE
BIOES	B-PER	I-PER	E-PER	O	O	O	S-GPE	O	B-DATE	E-DATE

* “Named entity” means proper names. However, in practice we train models to recognize whatever type of entity we need, not necessarily named ones.

Model and visualization: Spacy. Labeling schemes: Cho et al, 2013, Named Entity Recognition with Multiple Segment Representations

LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention

- LUKE is trained to predict randomly masked words and entities using a large amount of entity-annotated corpus obtained from Wikipedia.
- The model utilizes an entity-aware self-attention mechanism, which considers the type of the tokens (words or entities) when computing attention scores.

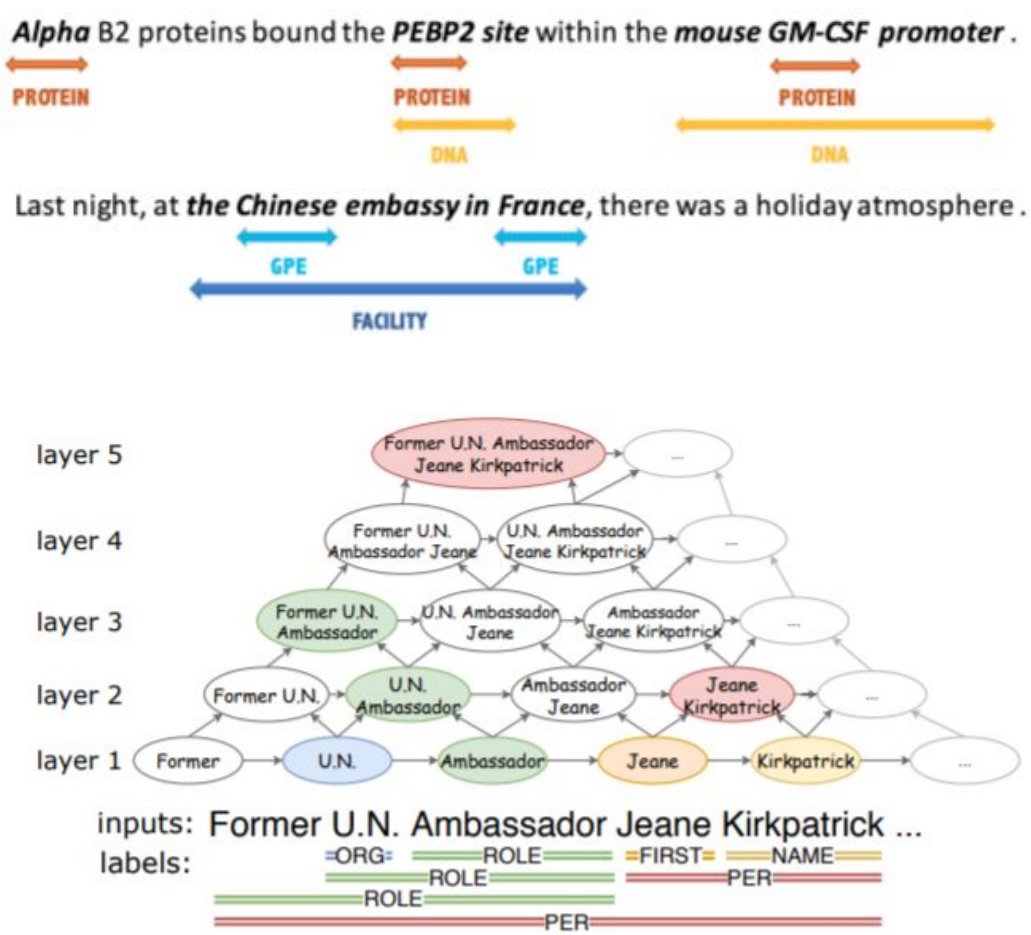


$$e_{ij} = \begin{cases} \mathbf{K}\mathbf{x}_j^\top \mathbf{Q}\mathbf{x}_i, & \text{if both } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are words} \\ \mathbf{K}\mathbf{x}_j^\top \mathbf{Q}_{w2e}\mathbf{x}_i, & \text{if } \mathbf{x}_i \text{ is word and } \mathbf{x}_j \text{ is entity} \\ \mathbf{K}\mathbf{x}_j^\top \mathbf{Q}_{e2w}\mathbf{x}_i, & \text{if } \mathbf{x}_i \text{ is entity and } \mathbf{x}_j \text{ is word} \\ \mathbf{K}\mathbf{x}_j^\top \mathbf{Q}_{e2e}\mathbf{x}_i, & \text{if both } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are entities} \end{cases}$$

Source: (1) https://arxiv.org/abs/2005.07150, (2) https://aclanthology.org/2020.acl-main.525/, (3) https://arxiv.org/abs/1910.11476

Recognition of nested or overlapping entities

- Sometimes, named entities nest or overlap, so the problem cannot be stated as *flat* sequence labelling
- Alternative formulations:
  - Scoring each span with a biaffine classifier (as in dependency parsing)[1]
  - Add extra layers ("pyramid") to predict entity labels for n-grams[2]
  - Re-formulate the problem as question answering, where entity type is the question[3]
    - Because same-type entities still can overlap, an additional layer for filtering candidate spans is required



Source: https://arxiv.org/pdf/2010.01057.pdf

Question answering

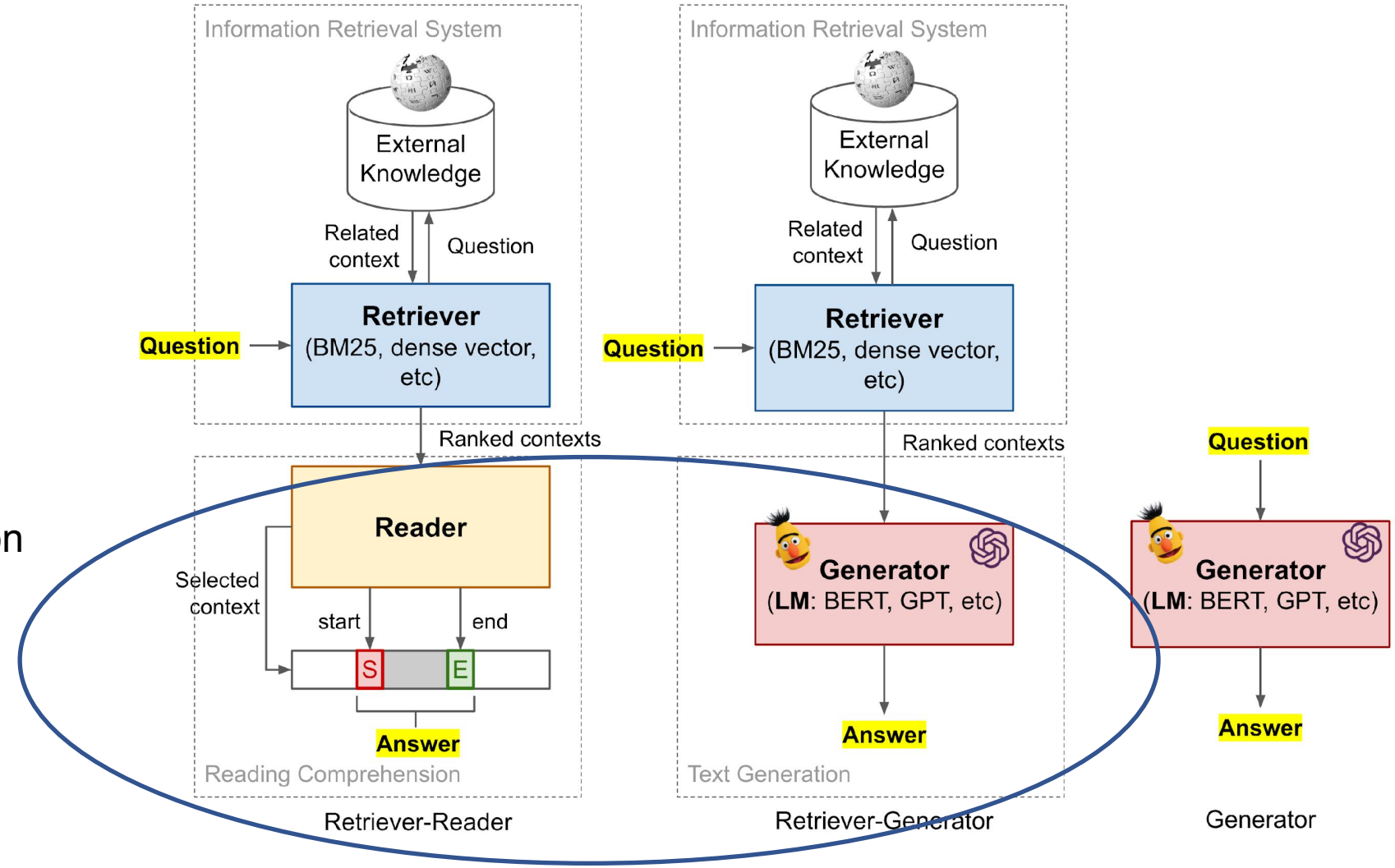
Typical QA problems and benchmarks

- Extractive QA
  - SQuAD: passage + question; the answer is within the passage
- Open generative QA
  - CoQA: passage + dialogue with questions; the answer is free-form text, but for each answer there is evidence within the passage
  - TriviaQA: question + long evidence documents; the answer is free-form text
- Closed generative QA
  - Just questions are given
- KBQA
  - Question + knowledge base, the answer should be selected or generated

Sources: https://github.com/danqi/acl2020-openqa-tutorial, https://huggingface.co/tasks/question-answering

Question answering system architectures

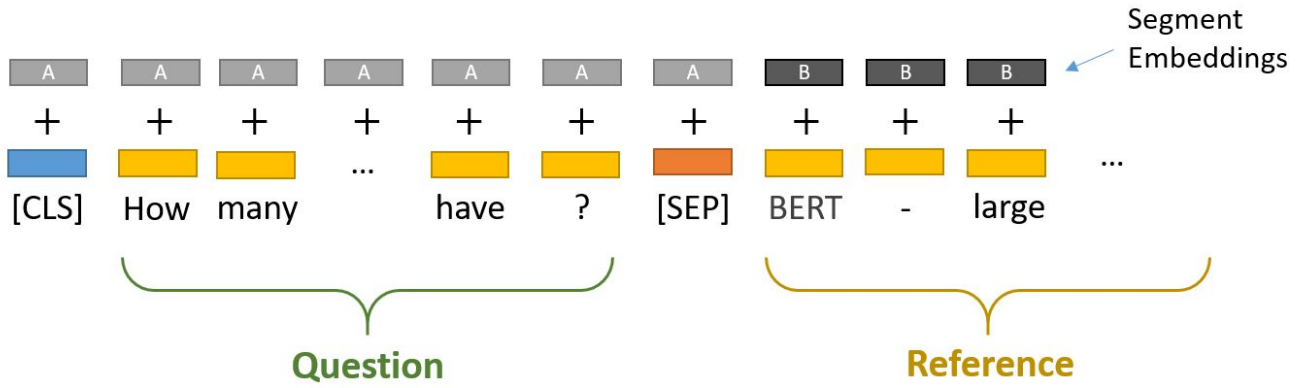
Reading comprehension



Source: https://lilianweng.github.io/posts/2020-10-29-odqa/

Extractive question answering

- Used for SQuAD-like tasks: given the question and the context, select the segment of the context which is the right answer
  - If there is no answer, select the first zero-length segment (i.e. [CLS] token)
- Inputs: the question + separator + the context
- Outputs: probabilities of each segment
- Evaluation: exact match or F1 over words



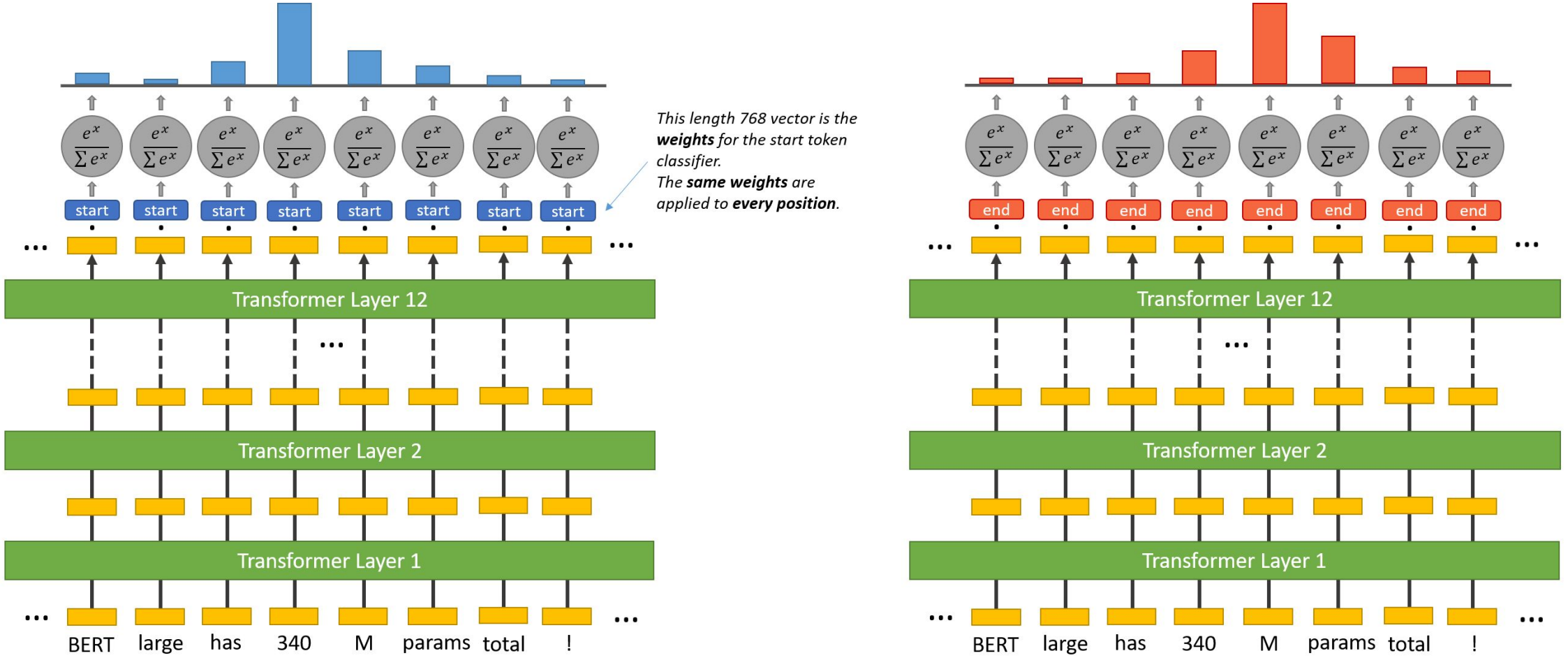
Question: How many parameters does BERT-large have?

Reference Text: BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

Source: https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/

Extractive question answering

The most likely answer is the span with the maximal product of start and end probabilities.



Summary

- Encoder-based models (such as BERT) are good for NLU:
  - Classification of texts, tokens, or relations between them
  - Encoding texts into informative vectors
- Many diverse NLP problems can be reformulated as sequence or token classification and solved by standardized models
- Systems based on transformer encoders are widely used for practical applications, such as information retrieval or chatbots